Введение в Greenplum

Раздаточный материал к 1-му уроку, посвященном введению в БД Greenplum.

|  |  |
| --- | --- |
| СОДЕРЖАНИЕ | |
| 1 | Понятие MPP |
| 2 | Архитектура Greenplum |
| 3 | Отдельные элементы сегментов и мастера |
| 4 | Отказоустойчивость |
| 5 | Источники информации (ссылки) |

Понятие MPP

* В системе SMP каждый процессор использует одни и те же ресурсы.
* В системе MPP каждый процессор имеет свои собственные выделенные ресурсы и ничего не использует совместно. Другими словами, система SMP имеет сильно связанные процессоры, а система MPP имеет более слабо связанные процессоры.

**Массово-параллельная архитектура** – это класс параллельных вычислительных систем, состоящих из множества узлов, где каждый узел представляет собой автономную, независимую от других единицу. Если применить это определение к области хранилищ данных, то лучше всего его смысл будет отражать термин «распределённые базы данных». **Каждый узел в распределенной базе данных представляет собой полноценную СУБД, работающую независимо от других.** Сама же распределенная база данных – это совокупность независимых, автономных узлов, связанных коммуникационной сетью.

Все данные в такой сети распределяются между узлами равномерно, т.е. каждый узел хранит свою, уникальную часть данных, логически, тем не менее, представляя единую базу данных. Для пользователя распределенная БД выглядит как единая целая, не разделенная на части база данных. **При обращении к распределенной СУБД запрос выполняется параллельно всеми узлами базы данных**, выполняя поиск и выборку только в своем, уникальном кусочке данных, что позволяет значительно повысить скорость доступа к данным. Преимущества такой архитектуры очевидны – это линейная масштабируемость, которая обеспечивает стабильные и предсказуемые параметры производительности и развитие системы.

**Преимущества MPP:**

* **относительная быстрота обработки больших объемов данных** при выполнении сложных SQL-запросов за счет распараллеливания операций и концепции Shared Nothing, когда каждый процессор имеет доступ только к своей локальной памяти и нет необходимости в потактовой синхронизации процессоров;
* **простота горизонтального масштабирования** до сотен узлов;
* **отказоустойчивость** за счет зеркалирования и резервирования;
* **линейный рост производительности.**

**Недостатки MPP:**

* **высокие требования к ресурсам ЦП**, памяти и жестким дискам, а также к сетевой инфраструктуре;
* **низкая производительность при большом объеме простых запросов**, выполняющих одну операцию, т.к. каждая транзакция на мастере порождает множество зеркальных транзакций на сегментах;
* **неоптимальное распределение сегментов** может негативно отразиться на производительности кластера при его расширении.

Архитектура Greenplum

**СУБД Greenplum представляет собой несколько взаимосвязанных экземпляров базы данных PostgreSQL,** объединенных в кластер по принципу массивно-параллельной архитектуры без разделения ресурсов (Shared Nothing). При этом каждый узел кластера, взаимодействующий с другими для выполнения вычислительных операций, имеет собственную память, операционную систему и жесткие диски.

Для повышения надежности к типовой топологии master-slave добавлен резервный главный сервер. В состав кластера Greenplum входят следующие компоненты:

* **Мастер-сервер** (Master host), где развернут главный инстанс PostgreSQL (Master instance). Это точка входа в Greenplum, куда подключаются клиенты, отправляя SQL-запросы. Мастер координирует свою работу с сегментами – другими экземплярами базы данных PostgreSQL. Мастер распределяет нагрузку между сегментами, но сам не содержит никаких пользовательских данных – они хранятся только на сегментах.
* **Резервный мастер** (Secondary master instance) — инстанс PostgreSQL, включаемый вручную при отказе основного мастера;
* **Сервер-сегмент**(Segment host), где хранятся и обрабатываются данные. На одном хост-сегменте содержится 2-8 сегментов Greenplum – независимых экземпляров PostgreSQL с частью данных. Сегменты Гринплам бывают основные (primary) и зеркальные (mirror).
* **Primary-сегмент** обрабатывает локальные данные и отдает результаты мастеру. Каждому primary-сегменту соответствует свое зеркало (Mirror segment instance), которое автоматически включается в работу при отказе primary. Производительность базы данных соответствует скорости самого медленного сервера. Рекомендуется, чтобы все хосты имели идентичные аппаратные ресурсы и конфигурации
* **Mirror segment instance** («зеркало») — инстанс Postgres, являющийся зеркалом одного из primary сегментов, автоматически принимает на себя роль primary в случае падения. GP поддерживает только 1-to-1 репликацию сегментов: для каждого из primary может быть только одно зеркало.
* **Интерконнект (interconnect)** – быстрое обособленное сетевое соединение для связи между отдельными экземплярами PostgreSQL.

В Greenplum каждая таблица представлена в виде (N+1) таблиц на всех сегментах кластера, где N – число сегментов + 1 таблица на мастере, где нет пользовательских данных. На каждом сегменте хранится 1/N строк таблицы. Логика разбиения таблицы на сегменты задаётся ключом (полем) дистрибуции, на основе которого любую строку можно отнести к одному из сегментов. Поскольку именно ключ дистрибуции задает распределение данных по сегментам кластера, выбирать это поле нужно по принципу равномерного распределения значений в нем.

**Master segment**

**Мастер — это место, где находится глобальный системный каталог**. Глобальный системный каталог представляет собой набор системных таблиц, содержащих метаданные о самой базе данных Greenplum.

**Мастер не содержит никаких пользовательских данных**; данные хранятся только на сегментах.

Мастер аутентифицирует клиентские соединения, обрабатывает входящие команды SQL, распределяет рабочие нагрузки между сегментами, координирует результаты, возвращаемые каждым сегментом, и представляет конечные результаты клиентской программе.

**Мастер взаимодействует с сегментами Greenplum следующим образом**:

1. Пользователь подключается к базе данных с помощью клиентских программ: psql или через API-интерфейсы типа JDBC и ODBC;
2. Мастер аутентифицирует клиентские соединения и обрабатывает входящие SQL-запросы. Каждый сегмент для обработки запроса создает соответствующие процессы;
3. После выполнения вычислений над локальными данными сегмент возвращает результаты мастеру;
4. Мастер координирует результаты от сегментов и представляет конечный итог клиентской программе.

Отдельные элементы сегментов и мастера

Мастер сегмент представлен следующими элементами:

**Query Parser** валидирует синтаксис, семантику и генерирует дерево запроса для Query Optimizer;

**Query Optimizer** – определяется план запроса, выбирается путь выполнения запроса с наименьшей стоимостью

* Генерируют планы запроса;
* Каждый план имеет стоимость;
* План с меньшей стоимостью передаётся диспатчеру.

**Query Dispatcher**

* Передаёт план запроса сегментам;
* Аллоцирует ресурсы на сегментах;
* Аккумулирует финальный результат для передачи клиенту

**Query Executor** (расположен на мастере и каждом из сегментов)

* Выполняет шаги плана;
* Обменивается промежуточными результатами с другими Query Executor (на других сегменах);
* Каждый QE запускает для запроса несколько воркеров;
* Одинаковые воркеры на сегментах – gangs.

**Интерконнект(ы)**

* Обменивается записями в промежуточных шагах;
* По умолчанию – своя реализация TCP поверх UDP, можно переключить на TCP

**Системный каталог**

* pg\_catalog – хранит в себе информацию об объектах базы;
* Дублируется на всех сегментах

**Distributed Transaction Manager**

* Мастер контролирует COMMIT/ABORT/ROLLBACK на сегментах;
* У каждого сегмента свои WAL, сегменты сами решают, когда коммитить и прерывать транзакции;
* Автокоммит включён.

Отказоустойчивость

**Зеркалирование мастера**

При необходимости можно развернуть резервную копию (или зеркало) главного мастера. Запасной мастер находится в режиме ожидания и **берет на себя функции мастера в случае, если основной хост становится неработоспособным**. Резервный мастер можно развернуть на назначенном хосте или на одном из хостов сегмента.

Резервный мастер содержит все необходимые для работы данные благодаря синхронизации между основным и запасным инстансами. Если основной мастер выключается из работы, процесс репликации журнала завершается, и администратор может инициализировать резервный мастер заново. Во время работы резервного мастера **реплицированные журналы используются для восстановления состояния основного мастера.**

Поскольку мастер не содержит никаких пользовательских данных, **необходимо синхронизировать только таблицы системного каталога**. Когда данные таблицы обновляются, изменения автоматически копируются на резервный мастер, и поэтому он всегда синхронизирован с основным.

Зеркалирование сегментов

При включенном зеркалировании система автоматически переключается на зеркальный сегмент, если основная копия становится недоступной. В случае если сегмент или хост выходит из строя, ADB может оставаться в рабочем состоянии при условии, что **все части данных доступны на оставшихся активных сегментах.**

Если мастер не может подключиться к сегменту, он отмечает в каталоге, что данный сегмент недействителен. **Сегмент остается недействительным и не работает до тех пор, пока администратор не возвращает его онлайн.** Процесс восстановления копирует пропущенные во время нерабочего состояния сегмента изменения.

При неподключенном зеркалировании в случае, если сегмент становится недействительным, **система автоматически отключается**. При этом администратору необходимо восстановить все сегменты прежде, чем продолжатся операции.

Источники информации (ссылки)

1. [Просто и доступно о аналитических БД](https://habr.com/ru/post/149641/)
2. [Что MPP-СУБД и как это работает](https://www.bigdataschool.ru/blog/what-is-mpp-greenplum.html)
3. [MPP Vs. SMP Database](https://www.techwalla.com/articles/mpp-vs-smp-database)
4. [История разработки и развития](https://www.bigdataschool.ru/wiki/greenplum)
5. [Зелено – не молодо: как устроена MPP-СУБД Greenplum](https://www.bigdataschool.ru/blog/greenplum-architecture.html)
6. [Greenplum DB](https://habr.com/ru/company/tinkoff/blog/267733/)
7. [About the Greenplum Architecture](https://docs.vmware.com/en/VMware-Tanzu-Greenplum/6/greenplum-database/GUID-admin_guide-intro-arch_overview.html)